下载APP

18 | 架构设计，专业分工和协作精神的体现

2020-12-04 乔新亮

乔新亮的CTO成长复盘 进入课程 >

讲述：乔新亮
时长 20:24 大小 18.69M

你好，我是乔新亮。今天，我想和你聊聊，关于架构设计的一些认知和体会。

作为技术人，最常接触的概念，恐怕就是架构设计了。即便是初出茅庐的新手程序员，可能也听说过 6 大设计原则与 23 种设计模式。因为，要成为管理者或技术专家，架构设计绝对是你绕不开的槛。

因此，关于架构设计的书和课程非常多，多到简直学不完。如果用我们专栏文章的形式来讲，写出一百篇文章都不为过。

但在今天，我们只聊一点认知，我认为，也是架构设计最核心的认知：**好的架构设计，就像优秀的组织协作一样，既能帮助 IT 系统做好各模块的专业分工，又能体现模块间的协作精神。**

认知讲完了，是不是感觉有些熟悉？

没错，其实在很多架构设计思想里，我们都能找到这句话的影子，比如“高内聚、松耦合”、“单一职责原则”、“接口隔离原则”，太多了。

你可能会想，老乔坑人呢，作为一个 CTO，在架构设计这一讲，就交付了一个这么基础的概念。

没错，这个概念确实很基础，但我想告诉你，其实架构设计，尤其是业务 / 应用架构的设计，原本就没有那么复杂。

工作了近 20 年后，我发现，如果一名架构师存在职业发展瓶颈，那么他的问题大概率是简单的设计方法没有执行好，而不是复杂的设计方法没学会。

复杂的学会了，简单的却没做好，听起来很奇怪吧？下面，我们详细聊聊，这究竟是怎么一回事。

从程序员到架构师，什么能力在提升？

2020 年，刚到彩食鲜的时候，我去审查团队技术需求的分配和实现，发现了一些不合常理的情况：对于部分业务流程改造工作，我认为很容易实现，团队却需要花费大量的时间才能落地。几天就能改好的代码，花掉一个月的时间都是有可能的。

有时，团队甚至会觉得，单是为了实现需求，就已经很吃力了，从整体的逻辑上看就很不合理。

仔细一查，我才发现问题的根源所在：表面上看，是逻辑不合理，实际上是架构设计不合理。

生鲜 / 电商类业务的架构一般会包括 OMS（订单管理系统）、WMS（仓库管理系统）、TMS（运输管理系统）等系统。其中 OMS 处理用户订单，负责调度；WMS 负责仓储管理；TMS 负责安排物流运输。可以看到，其实功能的划分很明确。

但很多业务在研发迭代时，并没有做过架构方面的考量，比如将与订单相关的逻辑直接放在 WMS 处理了：WMS 接到 1000 斤蔬菜的订单，检测仓库内只有 500 斤，于是发送请

求到采购系统，要求再采购 500 斤回来，然后将订单交付。

这样的逻辑能保证功能实现吗？当然可以，但长远看来，就会出现架构层面的设计问题 —— WMS 越来越臃肿，最终导致新需求的处理速度严重下降，影响业务的增长，这就是很多企业正在为之头疼的现实。

你可能会想，这个例子里的架构师真傻，连 OMS 都不知道。

真的是这样吗？事实上，类似的错误，很多企业每天都在犯，尤其是 IT 服务于自身生态的甲方企业。大家习惯于基于业务需求写代码，而不是基于架构设计写代码，反正来了需求先实现了再说，尤其是在业务刚刚起步时。

比如，在业界，你会发现一个很普遍的现象：许多企业投入大量的资金和人力进行 IT 建设，但每隔几年就要进行一次架构上的大调整，甚至直接推翻重写。

大部分人甚至已经习惯了这种行为，觉得重建很正常，而且是一名架构师能力的证明。

事实上，如果架构师严格按照“专业分工、充分协作”的思想去迭代需求，这样的重建根本就不应该出现。**一名优秀的架构师应该像个“隐形人”，似乎什么都没干，但其负责的架构就是能快速响应业务的需求。**

什么叫做“快速响应业务的需求”？就是说，在同样的业务复杂度下，在系统建设的不同阶段，可以使用相近的工作量完成需求。

那么，所谓的“专业分工、充分协作”，到底是做了什么呢？回到架构设计实践里，无非是一个先拆分再合并的 “V” 字型设计过程。

拆分是指将一个负责的功能按角色、按职责拆分，核心特征是单一模块的功能既单纯又简单。比如，要从零实现一个淘宝网，是相当复杂的事。但我们可以将其拆分成订单中心、用户中心、商品中心、库存中心等许多模块；订单中心还可以再拆分，比如拆分成订单创建、订单查询、订单履约等功能；如果实现仍然复杂，我们还可以继续拆分，直到能够用简洁优雅的代码去实现。

合并是指将类似的职责放在一个模块完成，抽象出可重用、复用的部分，提升业务响应效率。比如订单创建、订单查询都需要对数据库进行操作，那么与数据库的交互就应该统一封装。

抽象地看，架构是由元素（element）和关系（relationship）组成的。在架构设计中，那些稳定、可复用的部分应该变成组件或应用模块，对应着架构中的「元素」；而面向不确定性的设计，则需要变成协作方式，为可能的扩展作准备，对应着架构中的「关系」。

这就像一个足球队，有人司职前锋、有人司职中场、有人司职中后卫，分工明确才能组成一个有战斗力的队伍；中场可能回撤参与防守，后卫也可能前插参与进攻，这样才能对可能的变化作出响应。

如果什么设计都没有，就只能变成一群追着球玩的小孩。

那么「元素」是拆分得越细越好吗？当然也不是，5 个模块能搞定的功能，你非要写 100 个模块，只能是给自己徒增烦恼。

所以，从初级架构师到高级架构师，什么能力是一直存在并持续提升的？其实就是对复杂业务的拆分能力、对可复用部分的抽象能力、对拆分过程的颗粒度把握，以及对未来变化的考量和设计。如何让架构有足够的“应变能力”，则与架构师对业务的理解程度息息相关。

如果用一句话总结，那就是：**对架构层面的「专业分工」和「协作精神」的理解，是一名架构师的基础与核心能力。**

认知延伸：如何看待微服务和中台架构

你可能会想，老乔讲的对是对，可有什么用呢？

别急，我们通过前面讲的架构设计思想，反过来看看近几年大火的微服务和中台架构。

关于微服务，首先我们要知道，它的功能和数据处理是封装在一起的，这和 SOA 架构非常类似；其次，服务交互、服务治理、服务监控、服务隔离等很多基础性能力已经封装在框架中，可以让开发团队聚焦在功能实现上；再者，通过和 Kubernetes 集成，微服务的功能、数据、基础设施被再次封装，技术架构也再次进行了升级。

看得出来，在微服务框架下，技术架构部分的很多职责已经被抽象出来，并对框架进行了处理，隐含着相关理念的最佳实践。

同时，微服务也有一个很基本的原则：**让系统的分工更明确、责任更清晰**。所以你看，还是我们上面讲的那些内容。

对于业务侧架构师来说，即便是使用了微服务架构，工作还是会集中在架构设计方面，比如：业务功能如何进行归类。具体到微服务改造的过程中，就会碰见一个典型问题：微服务拆分的颗粒度怎么把控？微服务，英文 MicroService，划分粒度一定要很细吧？

其实这个问题压根没有统一答案，为什么？因为我们前面讲过了，即便没有微服务，架构师也要做功能的拆分，至于对颗粒度的把握，既受业务本身的特性影响，也受架构师的能力影响。

换句话说，一名对架构的“分工”和“协作”理解得足够好的架构师，很少会困惑于微服务拆分的颗粒度问题。因为从本质上来讲，这些都是一码事。从单体应用到各功能独立服务，其实是处于功能拆分的两个极端，但**架构设计本质上是一种“中庸思想”，如果单纯考虑功能设计，我们的目标只有一个：让架构响应业务调整的速度更快。**

那么架构如何才能实现更快的响应速度呢？就是在**保证各「元素」职责清晰的情况下，抽象出稳定的功能或组件，用业务流程去串联起来**。在一定时期内，一家企业的技术架构的功能或者组件基本是稳定的，而业务如何运转，却是动态变化的，甚至每周都在变。可以说，以不变应万变是架构设计的精髓。

这个设计思想又和中台架构如出一辙。企业搭建中台，最重要的目标之一，就是实现企业营收层面的“开源”，承担企业架构快速响应市场需求的任务。其关键词包括：消除烟囱、架构解耦、统一中台、服务重用……

没错，和我们前面谈的架构设计核心思想又是基本一致的。

关于中台，业界很多技术人都踩了不少坑。刚开始听说中台概念的时候，大家一拥而上，做得多了才发现，中台到处都是坑。以至于在当下，许多人对中台是嗤之以鼻的。

为什么呢？我认为很重要的一个原因，是大家忘了架构设计的“初心”，错认为中台是个全新的设计理念。

其实从架构的视角看，中台这个概念并不新鲜，它只是突出了“可重用部分的抽象”这部分工作。那么， 如果你的 IT 系统可复用部分不多、业务量不高，建设中台的意义何在呢？如果你的中台对业务没有帮助，建设中台的价值如何体现呢？

你看，时刻牢记架构设计的核心原则，能帮助我们更清晰、更透彻地看待当下的许多“时髦理念”。

当然，这里可不是说，中台的兴起，就是一群不专业人士的狂欢，绝对不是这样的。苏宁在 2012 年就开始中台建设，2013 年上半年完成。后来在利用中台接入天猫业务时，团队仅花了七天七夜的时间就完成了融合，速度非常之快。

关于企业数字化转型、中台建设，我在公众号里，曾经分享过一个系列，共 9 篇文章，用大概 4 万字左右的篇幅，描述了一个关于中台的“指挥官体系”，大家可以读一读，理解下中台和微服务设计的具体内容。

所以，无论是微服务还是中台，都有其巨大的实践价值，但二者只是架构设计核心原则的一种成熟的实践模式和承载方式，不是解决架构问题的“灵丹妙药”。

简单来说，如果你能按照架构设计的核心原则，做好模块拆分，那么微服务架构可以非常好地承载这种设计思想，为你提供服务治理、监控等一系列工具。但如果你在架构设计层面就做不好拆分，微服务也不能直接帮你解决颗粒度问题。

如果你问我，老乔，我没有“专业分工”的设计意识，也做不好功能的拆分，是不是用了微服务就搞定了？我只能回答，怎么可能呢？天底下没有这样的美事。

复杂架构设计如何落地执行

下面我们再来聊聊，复杂的架构设计究竟是如何落地的。

在 IBM 工作的那段时间，我曾经内部分享过一门专业架构师培训课程，包括为期 5 天的 Enterprise Architecture（企业架构）培训课程、为期 3 天的 Architecture Thinking

（架构思维）培训课程、为期 3 天的 Component Modeling（组件建模）培训课程，以及为期 3 天的 Operation Modeling （运营建模）培训课程。

我尽量用最简单的方式，将功能性架构设计中，最简单直接的方法和步骤抽象总结出来，分享给你，其中也包含了少数我们上面提到的内容。

1. 关键认知：在个人视角里，整个世界都可以按照确定性内容和不确定性内容做简单区分；
2. 架构将其抽象为元素和关系，元素对应着确定性内容的处理，是看待这个世界的稳定视角；关系对应着不确定性内容的处理，是看待这个世界的响应视角；
3. 人类的理解能力有限。包含内容过多的元素，会导致理解困难，需要将其拆分；
4. 同理，将元素归类的时候，也不能贪多，不然会导致理解困难。优秀的架构师，会将合理数量的元素进行归类，归类后的整体一般被称作 component 或 module，也可以直译为“组件”。比如，我们永远以“元素数量为 10 ”作为一个衡量点，只要一个组件包含的元素 / 职责超过 10 个，就要进行拆分；
5. 元素归类一般采用 “V” 字型分析法，即流程分解为功能，功能聚合为组件；
6. 如果组件明确了，意味着职责就建设好了，架构的元素（element）建设问题就解决了。组件对外暴露的能力，我们统一称之为服务；
7. 那么，架构的关系（relationship）建设问题该如何解决呢？稳定的关系，用来表达确定性的交互，使用 SOA 架构，做好服务调用就可以；不稳定的关系，用来表达不确定性的交互，使用 EDA 架构；
8. 在每一个新需求到来时，尤其是大版本的更迭，架构师需要介入产品经理和程序员的沟通中，判断新需求是否大幅增加了某些组件的复杂度，并决定是否调整组件职责，或对现有组件进行拆分。所以，从实际的业务发展来看，组件的数量是逐步增长的，如果一开始就很多，基本属于过度设计；如果业务复杂度已经翻倍了，组件数量却没变，基本属于缺乏设计。

那么以上就是落地复杂架构设计时的一些关键认知，要结合前文的阐述，多多体会。如果有疑惑，欢迎留言提问。

结语

今天我们聊了聊架构设计的核心理念：专业分工和协作精神，具体来讲，就是做好功能的拆分，抽象其中可复用的部分，并保留面向未来的扩展空间。

这里我们聊的仅限于功能型架构的设计，关于高性能、高可用、高并发、风险控制等非功能型架构设计，我们将在后续内容里逐渐展开。

我猜，即便看到这里 —— 这一讲的结尾，你可能仍然会想：这个理念是不是太简单了。

其实很多知识恰恰是这样：说简单也简单 —— 容易理解，也容易操作；说难也难 —— 即便是首席架构师、技术总监，可能也还是会被类似的问题困扰，重复犯类似的设计错误。

究其本质，在于我们要用发展的眼光，看待个人成长、架构设计这类相对宏伟的命题。就像我常常说的，要做时间的朋友。

做架构设计的同学都知道，罗马不是一天建成的。同样，对架构设计核心原则的理解，也不可能在一年、两年内达到满分，这种理解往往会随着技术人的成长而持续加深。

其实，人类在解决很多复杂问题时，都会采用类似的思维流程：**将复杂问题拆解为简单问题，逐一解决再合并，并将可复用的知识抽象，以实现举一反三**。我们今天所聊的架构设计，其实就是这种思想在软件设计层面的复现。

希望你也能将这个看似简单、实则复杂的知识点吃透、嚼烂，最终做到返璞归真，成为一个优秀的架构师或技术管理者。

我们下一讲再见。

提建议

© 版权归极客邦科技所有，未经许可不得传播售卖。 页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 17 | 架构决策，是技术管理者最重要的能力

精选留言 (11)

写留言

:)

2020-12-04

架构设计的终极目的： 快速响应业务需求，对业务达到可持续的稳定的支撑！
为了达到上述的目的，不同的时间，不同的地点，不同的环境，不同的发展阶段，都可能有不同的解决方案措施，但是别问白猫黑猫，抓到老鼠就是好猫。
之所以有各种各样的架构，很大程度是为了应对"复杂性"， 如果事情本身不复杂，为了架构而架构，就是舍本逐末。而影响到事物复杂性的因素主要有：事物数量多，事物联系...
展开

作者回复: 你好，:)
说的太好了

4

E

2020-12-04

乔老师这么多节课程其实一直在给我们布道“Trade-Off”的理念，要懂得前进，也要懂得刹车，不能陷入任何一种偏执的境地，随时提醒自己做这件事情的最初目的，不然容易走火入魔。

作者回复: 你好，E
谢谢分享，总结的特别好

2

Keith T.Maxwell
2020-12-04

这节内容我重复听了3个多小时。

展开

sotey
2020-12-04

乔老师，我其实对于技术管理有一点困惑，走上了管理岗位，也出了一些成绩。我个人认为从管理角度看技术，从价值层面看技术，其实我觉得我对技术的理解能力和实践能力都其实是提升了。在面对一线的技术问题，我往往会追求系统性的解决问题，也要求我的团队这样去解决。但是很多人却会认为我最近两年都在搞管理，不怎么搞技术了，认为我的技术已经不深入了。我又不能直接去抢了一线技术人员的活干，我怎么能消除别人对我...

展开

叶绘落
2020-12-04

阅读的时候总是容易发散文章中的内容，将其与已知的事物联系起来，所以看待事物总会有似曾相识触类旁通的感觉。

在乔老师文章中让我更加明确了这个的想法。

...

展开

Lemon
2020-12-04

颗粒度的把控和功能模块的划分是架构师的基本功，各种各样的架构和原则好比是武功的招式和心法，只有基本功扎实才能使用好各式武功，使产品能在风云变幻的市场上见招拆招。

gg
2020-12-04

架构还是一门权衡的艺术，需要根据业务场景选择合适的架构，同时还需要兼顾组织结构。

行与修
2020-12-04

以理论为基础，以实践为抓手，优秀架构师都是苦炼出来的！兵无常势水无常形，从事架构工作也不应给自己设限，多吸取本行业及其它行业的优秀实践经验，通用思维与专业深度两手抓两手硬！

Tony.xu
2020-12-04

受教了，挺精辟的，要是以后能开个架构课程就好啦，期待一下

glutton
2020-12-04

流程分解为功能，功能聚合为组件，提供的能力为服务
大道至简！架构拼的并不是新名词，是内功。
这篇会反复读。

展开

> 作者回复: 你好，glutton
> 流程分解为功能，功能聚合为组件，提供的能力为服务。
> 这盘文章的很关键的点，另外就是要不断权衡复杂度，确定组件的颗粒度，不断调整，持续改进。

K战神
2020-12-04

道和术，很多技术方案和技术难点的设计我们都会。

但是对于业务拆分，业务组合，

根据组织结构人员安排，能力重新洗牌这些模块和设计。

…

展开

作者回复: 你好，K战神

是的，讲架构设计的书籍多如牛毛，一点都不是问题。

最关键的是认知到了之后，要去实践，自己去领悟，很多架构师其实都苦过，然后领悟到其中的道理。